VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvovaní individuálni odborné praxe
Individual Professional Practice in the Company

2012 Pavol Repáň

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Pavol Repáň**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Stratech Software s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. RNDr. Petr Šaloun, Ph.D.**

Konzultant bakalářské práce: Ing. Petr Havrlant

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1 OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 2.5.2012

.....

Podpis

Na začiatku tejto práce by som rád poďakoval firme Stratech Software s.r.o., za možnosť vykonania odbornej praxe. Ďalej by som rád poďakoval pánovi Ing. Petrovi Havrlantovi a Petrovi Bystrzyckému za oboznámenie s chodom firmy, a odbornou pomocou pri vykonávaní pridelených úloh. Pánovi doc. RNDr. Petrovi Šalounovi, Ph.D. ďakujem za poskytovanie odborného dohľadu nad spracovaním tejto práce.

Abstrakt

Bakalárska práca vás oboznámi s mojimi skúsenosťami, ako študenta vykonávajúceho odbornú prax v spoločnosti Stratech Software s.r.o. Cieľom je popísať zameranie firmy a popis pracovného zaradenia študenta. V ďalšej časti je popísaný zoznam pridelených úloh, spolu so zvoleným postupom riešenia. Zoznámim vás s technológiami, ktoré som v priebehu vykonávania praxe používal, a popisujem teoretické a praktické schopnosti, ktoré som získal. V závere práce hodnotím dosiahnuté výsledky, ako aj schopnosti, ktoré mi pri riešení úloh chýbali.

Kľúčové slova:

MVVM, C#, Silverlight

Abstract

This bachelor thesis was written to give information about my experience as a student who performs a practice in Stratech Software Ltd company. The main aim is to describe the focus of the company and a description of work classification of the student. The next part describes a list of assigned tasks together with selected procedure of solution. The reader will be also acquired with technologies that I have worked with during my practice, together with a description of theoretical and practical abilities that I have obtained. The conclusion of the thesis evaluates achieved results but also abilities that were missing during problem solutions.

Key words:

MVVM, C#, Silverlight

Zoznam použitých skratiek a symbolov

.Net	– Development Framework
C#	– Programing language
Guid	– Global Unique Identifier
MEF	– Managet Extensibility Framework
MVVM	– Model View ViewModel pattern
Prism	– Composite pattern
RIA	– Rich Internet Aplication
SDK	– Software Development Kit
UI	– User Interface
TFS2010	– Microsoft Team Foundation Server 2010
WCF	– Windows Communication Foundation
XAML	– Extensible Application Markup Language

Obsah

1. Úvod

Bakalárska práca popisuje absolvovanie odbornej individuálnej praxe v spoločnosti Stratech Software s.r.o. V práci sa venujem bližšiemu oboznámeniu so spoločnosťou v ktorej som prax vykonával. Popisujem ju ako z hľadiska odbornosti, tak aj z hľadiska pôsobenia v rôznych sférach trhu zo softvérovými produktmi. Stratech je softvérová firma, ktorá používa na vývoj aplikácií najnovšie technológie a návrhové vzory, čo pre mňa ako študenta predstavuje určitú výhodu pri profesnom uplatnení po skončení štúdií.

Ďalšia časť práce je venovaná stručnému popisu projektu na ktorého vývoji som sa podieľal, ako aj oboznámenie s technológiami s ktorými som pracoval pri vykonávaní odbornej praxe. Táto časť taktiež obsahuje zoznam pridelených úloh a ich riešeniami na ktorých som pracoval. Tento zoznam samozrejme nie je kompletný, obsahuje úlohy ktorých riešenie bolo komplexnejšie, časovo náročnejšie a sú z hľadiska implementácie komplikovanejšie ako rôzne menšie úlohy, ktoré mi boli pridelené. Kompletný zoznam úloh je pripojený v prílohe č.1. Keďže v spoločnosti Stratech sa pracuje pomocou agilnej metodiky Scrum, tak postup vo vývoj je veľmi dynamický. Prideľovanie úloh je priebežné, a v týchto úlohách sú odzrkadlené všetky aktuálne potreby implementácie. Praktické skúsenosti s prácou na zdieľanom kóde využívajúc TFS 2010, alebo zapojením sa do tímového agilného vývoja som nadobudol správne návyky, s ktorých môžem ťažiť do budúcnosti.

Posledná časť bakalárskej práce je venovaná prínosom odbornej praxe pre mňa, ako študenta, kde popisujem teoretické znalosti nadobudnuté v priebehu štúdia, ktoré som využil pri práci v spoločnosti Stratech. Taktiež je potrebne popísať aj nadobudnuté znalosti v priebehu praxe, či znalosti ktoré mi chýbali alebo neboli rozvinuté na potrebnej úrovni.

2. Predstavenie firmy Stratech Software

Stratech Software je medzinárodne aktívna spoločnosť, s hlavným sídlom v Enschede v Holandsku a dcérskymi pobočkami vo Francúzku a Česku, v ktorej pracuje celkovo viac ako 60 zamestnancov. Spoločnosť vznikla v roku 1989 a špecializuje sa na vývoj, predaj a implementáciu užívateľsky priateľského (user-friendly) softvéru. V súčasnej dobe Stratech naplno pracuje na migračnom procese, v ktorom aplikácie vyvinuté v predchádzajúcich rokoch založené na platforme Delphi prechádzajú na platformu .Net. Spoločnosť Stratech má záujem o inovatívne prístupy k riešeniam svojich produktov, a preto používa najnovšie technológie a návrhové vzory. Zameriava sa na upravovanie svojich produktov, presne na mieru každému klientovi. Vyvinuté produkty obsahujú požadovanú funkcionalitu, ktorú je možné rozšíriť o nezávislé moduly a tým upraviť poskytovaný produkt presne podľa potrieb klienta. [1]

Obrázok 1: Logo firmy Stratech Software

2.1. Oblasti pôsobenia

Stratech sa špecializuje na vývoj a implementáciu vysoko kvalitného softvéru pre konkrétne oblasti záujmu. Základom vyvinutých produktov a služieb je veľmi dôkladná znalosť v oblasti technológií a špecifických priemyslových oblastí. Spoločnosť udržuje rozsiahle kontakty s firmami a organizáciami pôsobiacimi v týchto oblastiach, čo prináša spätnú väzbu (feedback) potrebnú k vývoju efektívnejších a aktuálnejších riešení. Toto umožňuje okamžite reagovať na vývoj trhu a poskytnúť zákazníkom softvér na vysokej úrovni. [1]

Stratech je predným dodávateľom pre nasledujúce priemyselné odvetvia:

- rekreačný priemysel;
- logistika a priemysel;
- verejná sféra;
- výskum a vývoj.

2.1.1. Rekreačný priemysel

Stratech úspešne spolupracuje na automatizácií s viac ako 250 partnermi pôsobiacimi v rekreačnom priemysle. Základom týchto produktov je proces rezervácie a možnosti rozšírenia o správu informácii o klientoch, vytváranie a plánovanie denných aktivít, a taktiež možnosť spravovať a udržovať poskytovaný softvér. [1]

2.1.2. Logistika a priemysel.

Stratech je expert v automatizácii pre logistiku a priemyselne odvetvia. Od roku1989 získava rozsiahle znalosti o všetkých aspektoch kompletného spracovania dokumentov pre medzinárodnú prepravu, a to aj v oblasti colných záležitosti, a najrôznejších poplatkoch spojených s dopravou. Taktiež sa spoločnosť Stratech podieľa na rokovaniach priemyslových organizácii. Tieto skúsenosti sa odrážajú na kvalite a komplexnosti produktov určených tomuto odvetviu. [1]

2.1.3. Verejný sektor

Stratech využíva dlhoročnú spoluprácu s partnermi pôsobiacimi vo verejnej správe k tomu, aby zo zložitých procesov vytvorila jednoduché riešenia. Riešenia vznikajú za pomoci postupného zjednodušovania manažmentu jednotlivých časti. V tomto sektore poskytuje produkt, ktorý nepretržite sleduje efektivitu reintegračného procesu u zákazníka, a produkt pomáhajúci k zefektívneniu využívania časového rozpočtu klienta. Spoločnosť poskytuje nastroj na realizáciu politických cieľov v obciach, za pomoci efektívneho využívania dotácií a taktiež produkt obstarávajúci náležitosti súvisiace s čerpaním European Social Fond(ESF). [1]

2.1.4. Výskum a vývoj

Spoločnosť Stratech plánuje v roku 2012 poskytnúť novo vytvorený produkt RD-Process pilotným zákazníkom. Tento systém slúži na riadenie produktového manažmentu, a cieľová oblasť záujmu sú stredne veľké organizácie s počtom zamestnancov nie vyšším ako 500. Tento produkt zvýši efektivitu a prehľadnosť u produktov, ktoré sa rozhodne zákazník vytvoriť. Jedna sa o obecný systém ktorý je možné použiť ako na vývoj softvéru tak na vývoj konkrétneho fyzického produktu alebo služby.

2.2. Česka pobočka Stratech Software s.r.o.

Spoločnosť Stratech Software s.r.o. je novou spoločnosťou s výraznou podporou a získaným know-how od svojej materskej spoločnosti, ktorá sa zaujíma vývojom nových softvérov. Jej hlavným cieľom je vývoj, predaj, implementácia a údržba flexibilného štandardného softvéru pre optimálnu podporu organizácii R&D. Produkty budú nasadené v celosvetovom meradle a podporujú celkový proces vývoja a úpravy produktov. V súčasnej dobe Česka pobočka zamestnáva 10 zamestnancov, ale do budúcnosti predpokladá, že dôjde k rozširovaniu počtu pracovníkov v tíme. Hlavým cieľom spoločnosti sú činnosti súvisiace so spracovaním dát, hostingom a činnosťami súvisiacimi s webovými portálmi, alebo poradenstvom v oblasti informačných technológii.

2.3. Popis pracovného zaradenia študenta

Na vstupnom pohovore som sa zoznámil s tím lídrom a mojim budúcim vedúcim práce. Bol som oboznámený so základnými informáciami a technológiami používanými v ostravskej pobočke spoločnosti, a taktiež ma pozvali na druhé kolo pohovoru. Na tomto pohovore som sa zoznámil s ďalším študentom z VŠB-TUO, ktorý mal rovnaký záujem o vykonanie praxe vo firme Stratech Software s.r.o. Druhého kola pohovoru sme sa zúčastnili spoločne, a bol vedený manažérom z holandskej pobočky spoločnosti. Pri pohovore sme prebrali naše skúsenosti s programovaním, ako aj časový harmonogram a celkovú dĺžku nasej individuálnej praxe. Prvé dni mojej praxe som bol pridelený do tímu, ktorý vyvíjal produkt RD-Process. V tíme som bol pridelený k vývoju na užívateľskom rozhraní. Postupom času sa zmenila štruktúra tímu v ostravskej pobočke, a bol som zapojený do kompletného vývoja, zahŕňajúceho UI ako aj komunikačnú a biznis logiku pozadia aplikácie.

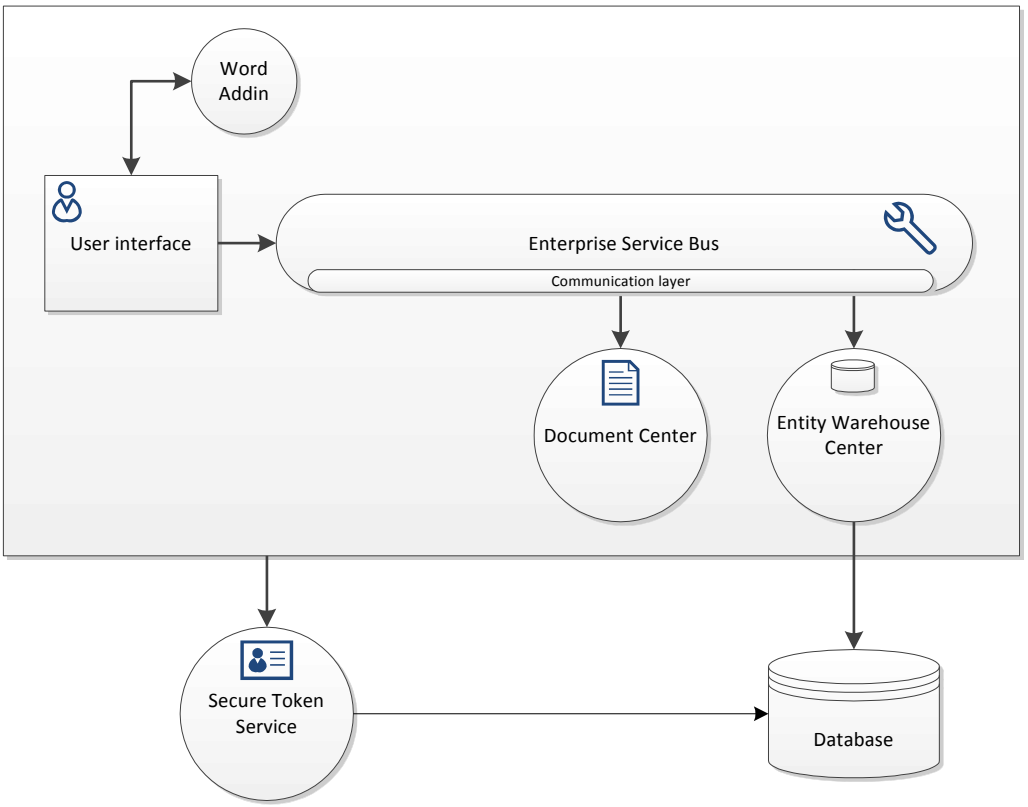
3. Popis vyvíjaného projektu

Prodigy platforma poskytuje súbor základných prostriedkov, z ktorých môže byť výsledný produkt zložený. Tieto základné prostriedky sú formované ako súbor modulov, ktoré môžu byť dopĺňané o novo vyvinutý produkt, či rozšírenie existujúceho produktu, tak aj pridaním ďalších modulov.

Existuje silná spätná väzba medzi základnými modulmi a produktmi. Používanie základných modulov je sledované a výsledky sa vracajú späť do vývojového procesu. V dôsledku toho sú základné moduly všeobecnejšie a majú väčší potenciál znovu využitia pri vývoji budúcich produktov. Aby sa tato vlastnosť dala naplno využiť, musíme mať podporu silného manažmentu, ktorý je schopný vyvíjať a udržiavať súbor základných modulov. Manažment musí taktiež urýchliť zmenu pohľadu na vývoj nového produktu za pomoci existujúcich základných modulov. Buď musí prispôsobiť nový produkt existujúcim modulom, alebo aktualizoval potrebne moduly aby odrážali aktuálne potreby trhu. [2]

3.1. Popis architektúry Prodigy

Platforma Prodigy je častou Prodigy Software Product Line (SPL). Skladá sa z niekoľkých centier, ktoré dohromady poskytujú všetky potrebné funkcie produktového portfólia Stratech. Kompletný pohľad na Prodigy platformu zobrazuje obrázok č.3. Tri hlavné komponenty platformy sú User Interface (PUI), Enterprise Service Bus (PSB) a Entity Warehouse (PEW). Tieto tri centrá poskytujú základné funkcie systému. Obrázok znázorňuje aj dve ďalšie centrá, ktoré tiež hrajú dôležitú úlohu v architektúre, ktorými sú Secure Token Service (STS) a Document Centrum (PDC).



Obrázok 3: Prodigy platforma.

Prodigy User Interface.

PUI -je súčasťou Prodigy platformy, ktoré je najviac viditeľné z pohľadu bežného používateľa. Poskytuje (user-frendly) rozhranie pre interakciu s používateľmi systému. Je to časť aplikácie, ktorá slúži používateľovi na výmenu informácií so systémom v oboch smeroch. Užívateľské rozhranie neobsahuje žiadnu biznis logiku, a je zodpovedné len za sprostredkovanie komunikácie medzi užívateľom a Enterprise Service Bus. PUI reprezentuje jeden cely projekt na Team Foundation Serveri, v ktorom je obsiahnutá prezentačná vrstva Prodigy architektúry.

Prodigy Enterprise Service Bus

PSB je zodpovedná za celu biznis logiku v systéme, rovnako ako aj komunikáciu medzi ostatnými Prodigy centrami a v neposlednom rade aj komunikáciou a používanie externých služieb 3. strán. PSB je taktiež reprezentovaná jedným projektom na TFS, ktorý obsahuje celu communiction / service vrstvu a vrstvu v ktorej je obsiahnutá celá biznis logika.

Prodigy Entity Warehous

PEW je zodpovedný za všetku komunikáciu s databázou. Zodpovedá za základné operácie nad databázou, ako sú select, insert, update, delete a operáciami nad dátami, ako je načítanie a ukladanie. K dátam majú prístup iba služby, ktoré prechádzajú cez PEW s výnimkou STS, ktorá používa prístup k dátam na autentizáciu a nasledovnú autorizáciu používateľa. PEW poskytuje dáta alebo dátové abstrakcie Prodigy architektúry a je reprezentovaný samostatným projektom.

Prodigy Secure Token Service

Úloha STS je dvojaká, slúži ako licenčný manažér a zaisťuje bezpečnosť celej architektúry. STS udržuje licencie všetkých zákazníkov a zabezpečuje, že zákazníci nie sú schopní používať akékoľvek produkty alebo moduly, na ktoré nemajú licenciu. Taktiež uchováva informácie o počte súčasne pracujúcich užívateľov a porovnáva ich s počtom vystavených licencií pre týchto užívateľov, čim zabezpečuje neustálu kontrolu.

Druhou úlohou STS je, poskytovanie bezpečnostného okruhu (token) používateľom zákazníckej spoločnosti. Tento token môže byť použitý pri procese autentizácie a autorizácie vo všetkých centrách a službách poskytovaných platformou Prodigy, ale aj mimo nej.

STS je samostatný projekt, ktorý realizuje zabezpečenie skrz celu architektúru Prodigy.

Prodigy Dokument Center

PDC poskytuje zázemie pre správu dokumentov v rámci platformy Prodigy. Je schopné ukladať a načítať dokumenty a šablóny, ktoré môžu byť použité v iných centrách. [2]

3.2. Popis produktu RD-Process

Produkt RD-Process je určený na pomoc spoločnostiam s vývojom ich vlastných výrobkov a je vyvinutý na platforme Prodigy. Je zameraný na zapracovanie požiadavkou (features), ktoré môžu pochádzať z rôznych miest, ako sú napríklad požiadavky trhu alebo jeho segmentu, organizácie, dodržanie zákonov, požiadavky na produkt alebo produktovú komponentu či projekt. Tieto požiadavky môžu byť rôznych typov ako biznis požiadavky, funkčné alebo nefunkčné požiadavky atď. RD-Process je systém vyvinutý k celkovej podpore riadenia produktu (Produkt management), ktorý podporuje výskum a vývoj produktu a pomáha

riadiť celý vývoj produktu od fázy myšlienky, nápadu cez fázu plánovania až do fáze vyvinutia a dokončenia.

RD-Process sa skladá zo siedmych hlavných častí, ktoré sú kombináciou základných modulov poskytovaných platformou Prodigy, ako aj modulov vyvinutých pre tento produkt. Základný produktový balík obsahuje moduly Glossary, Documentation, Product, Stakehoder, Project, Maintenance a Support tables. K tomuto základnému balíku sa dajú pripojiť aj ďalšie moduly ako Customer Relation Module (CRM) alebo Human Resources Module (HRM).

Glossary je modul zhromažďujúci a uchovávajúci slová a skratky, používané pri vývoji produktu. Je to určitý druh slovníka, ktorý je špecifický pre každú firmu a každý produkt.

Documentation je časť systému, ktorá uchováva všetku dokumentáciu potrebnú k výskumu a vývoju produktu, ako sú dokumenty typu technical design, functional requirements, rôzne výkresy, návody, príručky a schémy potrebne k vývoju produktu.

Product je časť určená na kompletne definovanie vyvíjaného produktu. Skladá sa z produktovej komponenty a štandardnej komponenty. U oboch sa definujú podrobne vlastnosti ako sú produktové požiadavky, use case a aktivity diagramy, ako aj popisy jednotlivých časti a pripadné problémy, ktoré môžu nastať pri vývoji.

Stakehoder je časť v ktorej sa definujú externé zásahy do vývoja produktu, či už sú to osoby mimo spoločnosť, tak to môže byť aj trh, zákon alebo iná organizácia atď.

Projekt je modul, ktorý definuje určitú časť produktu na ktorej sa aktuálne pracuje. Je rozdelený na dve časti, prvá časť je RD Projekt, ktorý sa vyvíja v rámci spoločnosti a druhá časť je Stakeholder Projekt, na ktorého vývoji sa podieľajú aj externý pracovnici, organizácie atď.

Moduly *Maintenace* a *Support tables* slúžia na údržbu a definovanie rôznych výberových kritérií pomocou ktorých sa kategorizujú projekty a produkty. Tieto kategórie sú napríklad rôzne fázy vývoja produktu, klasifikácie produktu, statusy a typy projektu atď.

3.3. Popis používaných technológií

.Net Framework

.NET Framework je platforma spoločnosti Microsoft, ktorá poskytuje možnosti vývoja, nasadenia a používania desktopových, webových či mobilných aplikáciách a webových služieb. Skladá sa z dvoch hlavných častí. Prvá Common Language Runtime (CLR) s implementáciou virtuálneho stroja, ktorý poskytuje dôležité služby, ako je zabezpečenie a správa pamäte. Druhá časť je Base Class Library, ktorá ponúka sieťovú komunikáciu, prístup k dátam, vývoj webových aplikácií, pripojenie k databáze, numerické algoritmy, užívateľské rozhranie atď. Tato platforma bola vyvinutá primárne pre aplikácie určené pre operačný systém Microsoft Windows. [3]

Microsoft C#

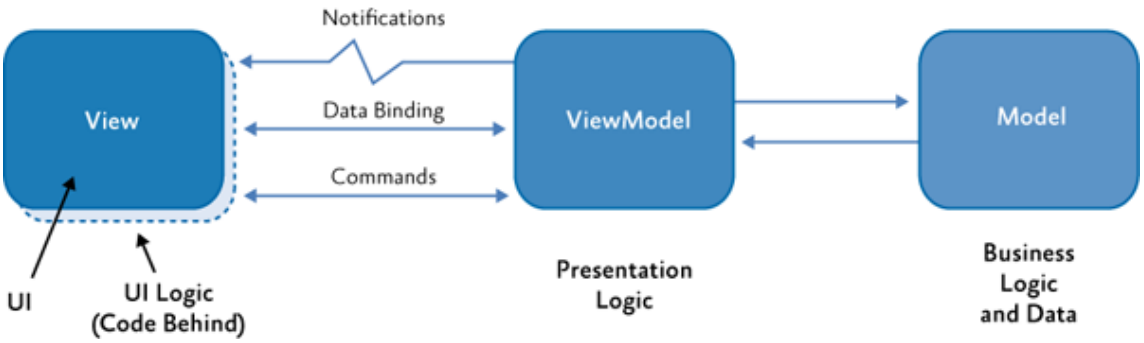
C# je programovací jazyk, ktorý je určený k vývoju rôznych aplikácií. Je platformovo neutrálny, ale bol navrhnutý aby efektívne spolupracoval s platformou Microsoft .Net. Je to univerzálny, jednoduchý, silný, typovo bezpečný, a objektovo orientovaný programovací jazyk. Veľké množstvo noviniek v C# 4.0 umožňuje rýchly vývoj aplikácií. Visual C# je implementácia jazyka C# spoločnosťou Microsoft. Visual Studio podporuje Visual C# s plnohodnotným editorom kódu, prekladačom, výkonným a ľahko použiteľným debuggerom a množstvom iných nástrojov. Platforma .Net poskytuje prístup k službám operačného systému a iným užitočným triedam, ktoré urýchľujú programovanie. [4]

Microsoft Silverlight

Silverlight je aplikačná platforma vytvorená spoločnosťou Microsoft, ako konkurencia technológií Adobe Flash, postavená na .Net Framework. Je určená na vytváranie a poskytovanie novej generácie práce s médiami a bohatými interaktívnymi aplikáciami (RIA) na webe. Taktiež existuje možnosť vytvárať Silverlight aplikácie, ktoré sú spúšťané mimo prehliadač na používateľskom desktope a v neposlednej rade môžeme použiť Silverlight na vývoj aplikácií pre Windows Phone. Používateľské rozhranie sa typicky definuje pomocou deklaratívneho programovacieho jazyka XAML, ktorý okrem vzhľadu UI môže popisovať napríklad aj animácie a väzby na dáta, avšak tato logika sa nachádza v code-behinde súbore. [5]

Model View ViewModel (MVVM) pattern

MVVM vzor pomáha čisto oddeliť obchodnú a prezentačnú logiku aplikácie od používateľského rozhrania. Udržanie čistého oddelenia aplikačnej logiky a používateľského rozhrania pomáha riešiť mnohé problémy vývoja a návrhu. Pomáha aplikáciu oveľa jednoduchšie testovať, udržiavať a rozvíjať. To môže tiež výrazne zlepšiť znovu použitie implementovaného kódu, čím umožňuje vývojárom a návrhárom používateľského rozhrania ľahšie spolupracovať pri rozvoji jednotlivých častí aplikácie.



Obrázok 4: MVVM pattern

MVVM definuje, že UI aplikácie rozdeľuje prezentačnú a biznis logiku do oddelených tried:

- **View** – definuje štruktúru a vzhľad toho čo užívateľ vidí na obrazovke. V ideálnom prípade code-behind obsahuje iba konštruktor, ktorý volá metódu InitializeComponent(). V niektorých prípadoch však môže obsahovať implementovanú logiku zložitých vizualizácií, ktoré je neefektívne vyjadriť pomocou XAML.
- **View Model** – obsahuje prezentačnú logiku a dáta potrebne na zobrazenie vo View. Nemá k dispozícií priame informácie o implementácii, alebo type View. View Model implementuje vlastnosti a príkazy, potrebne k tomu aby ich View dokázalo zobrazovať a upozorňovať na ich zmenu pomocou upozorňovacieho eventu. Vlastnosti a príkazy, ktoré udržiava View Model, definujú funkcionalitu celého používateľského rozhrania.
  V tejto časti môžu byť definované logické stavy, ktorými View mení vizuálnu časť UI. View Model definuje príkazy a akcie odzrkadľujúce zásahy používateľa v UI. Napríklad keď View Model obsahuje potvrdzujúci príkaz, ktorým užívateľ odošle zadané dáta na webovú službu alebo do dátového úložiska.

- **Model** – obsahuje celú biznis logiku a dáta. Biznis logika obstaráva manažment nad aplikačnými dátami, ktoré sú validované a pripravené na znovu použitie alebo uloženie. Model má implementovanú aj prístupovú logiku k uloženým dátam, ako aj používanie Repositories, alebo iných služieb. Často sa používa možnosť vygenerovania Modelu spolu s prístupom k dátovej vrstve pomocou ADO.NET Entity Framework, WCF Data Services alebo WCF RIA Service. [5]

Windows Communication Foundation (WCF)

WCF je SDK určený k vývoju a nasadeniu služieb v systéme Windows. WCF je implementáciou spoločnosti Microsoft, ktorá zahŕňa sadu priemyselných štandardov definujúcich interakcie služieb, typovú konverziu, klasifikáciu a manažment rôznych protokolov. Toto umožňuje súčinnosť medzi službami, ktoré používali rôzne staršie technológie. WCF tieto staršie technológie, ako je MSMQ alebo WSE, zjednocuje a tým uľahčuje celkový vývoj služieb. Základ použitia WCF je v aplikáciách, ktoré sú postavene na architektúre Service-Oriented Achutekture (SOA). Veľká výhoda WCF je v možnosti jednoducho nastaviť bezpečnosť a ďalšie vlastnosti služby. [6]

Microsoft Prism pattern

Prism je framework poskytujúci návod a pomoc pri návrhu a vývoji jednoduchej, flexibilnej a ľahko udržateľnej Silverlight RIA, Windows Phone 7, alebo WPF desktopovej aplikácie. Využíva návrhové vzory, ktoré stelesňujú dôležité architektonické princípy, ako je oddelenie zodpovednosti (separation of concerns) a voľná väzba (loose coupling). Prism pomáha navrhovať a vytvárať aplikácie pomocou voľne prepojených komponent, ktoré sa môžu vyvíjať samostatne, ale môžu byť veľmi ľahko a bez problémov integrované do celkovej aplikácie. Takéto aplikácie sú často označované ako kompozitné aplikácie. [7]

Managed Extensibility Framework (MEF)

MEF zjednodušuje rozšíriteľnosť aplikácií. Je to kompozičná vrstva platformy .Net 4.0, ktorá zlepšuje flexibilitu, udržateľnosť a testovateľnosť veľkých aplikácií. MEF sa používa na začlenenie technológii 3.stran, ale prináša aj výhody pri používaní voľno väzobných architektúr a nezávislých moduloch v aplikácií. [7]

3.4. Pridelené úlohy

Prvé dni vo firme boli venované integrácii a zaškoleniu. Spolu s vedúcim sme vytvárali používateľské účty, prihlasovali sa do domén a inštalovali softvér potrebný k práci na projekte Prodigy. Po nastavení prostredia som bol oboznámený s TFS2010, a začala moja práca na školiacom MVVM tutoriále so študovaním architektonických a programovacích príručiek. Po konzultácii tutoriálu s vedúcim mi bolo ozrejmené základné fungovanie vyvíjaného projektu.

Prvou pridelenou úlohou bolo doplniť lokalizáciu do používateľského rozhrania produktu RD-Process. Táto úloha mi pomohla sa zorientovať v projekte a taktiež ma naučila rozoznávať jednotlivé časti XAML súborov. Keďže táto lokalizácia sa dotýkala iba view vrstvy MVVM, ktorá je implementovaná v jazyku Silverlight, mal som možnosť sa bližšie zoznámiť s týmto jazykom a jeho štruktúrou.

Nasledujúce úlohy, ktoré mi boli pridelené mali charakter malých zmien v implementácii, či už na strane UI alebo logiky. Tieto zásahy do vývoja mi pomáhali lepšie pochopiť väzby medzi časťami kódu. Tým sa mi rozširoval celkový prehľad v komplikovanej štruktúre implementácie architektúry Prodigy a produktu RD-Process.

Po nadobudnutí potrebných znalostí a skúseností mi bola pridelená úloha implementovať časť Classification v Maintenace. Jedná sa o časť, ktorá umožňuje navrhnúť vlastnú štruktúru rozdelenia vyvíjaných produktov. Po vytvorení stromovej štruktúry ProductClassification je možné tento návrh používať pri kategorizácii každého vyvíjaného produktu. Táto úloha obsahovala vytvorenie a udržanie stromovej štruktúry pomocou nového prístupu k používaniu ViewModelu pod komponentom RadTreeView. Vytvorenie UI podľa návrhu spolu s kompletnou funkcionalitou Add, Save, Delete a Detail View, ktoré sú úzko spojene s komunikáciou a zmenami nad databázou.

Po dokončení Classifikation som dostával úlohy, ktoré sa týkali chýb zistených pri testovaní mojej implementácie, ako aj opravovanie a riešenie problémov spojených s testovaním produktu RD-Process. V poslednej časť môjho pôsobenia v spoločnosti Stratech som pracoval väčšinou na úlohách súvisiacich s odstránením nepoužívaných časti kódu, alebo s modifikáciou zastaraných funkcií pomocou novších, efektívnejších, prehľadnejších implementácií v rámci architektúry Prodigy. V tejto časti bakalárskej práce sa nezaoberám všetkými úlohami, ktoré mi boli pridelené, avšak kompletný prehľad odpracovaných dní a úloh je pripojený v prílohe č. 1.

3.5. Popis riešenia vybraných úloh

Na začiatku riešenia prvej úlohy, týkajúcej sa lokalizácie RD-Processu, som mal základný prehľad o štruktúre XAML ako aj o deklaráciách rôznych objektov pomocou jazyka Silverlight. K príprave slúžila aj príručka umiestnená na internom serveri, ktorá bola napísaná kolegami z ostravskej pobočky vyvíjajúcimi modul Localization Manager. Tento modul funguje na základe porovnania výrazu, ktorý do neho vstupuje, s výrazmi uloženými v tabuľke prekladov do požadovaného jazyka. Pokiaľ sa takýto vyraz v tabuľke nachádza tak ho Localization Manager automaticky prepíše do požadovanej jazykovej variácie, a ak sa v tabuľke nenachádza, tak odstráni podčiarkovník a zobrazí cely vyraz aj s prefixom. Výrazy používané k prekladu majú tvar prefix_prekladanyVyraz. Prefixy u týchto výrazov určujú k čomu sa daný vyraz v UI používa, napríklad prefix title_ znamená, že vyraz za prefixom sa používa iba ako zobrazovacia informácia, prefix act_ sa používa pri výraze označujúcom akciu ako act_Cancel.

Prvý krok bolo prepnúť si UI do holandského jazyka, aby som moje zmeny spozoroval okamžite. Pri nastavení NL jazyka sa všetky textové elementy na UI prepísali holandskými výrazmi. Ako ďalší krok som postupne prehliadol všetky View triedy v ktorých som prepisoval elementy ako Label, GroupBox, TextBlock alebo DataGridView.

```
<telerik:GridViewDataColumn Header="Name"  DataMemberBinding="{Binding Name}" IsReadOnly="True">

<telerik:GridViewDataColumn DataMemberBinding="{Binding Name}" IsReadOnly="True">
   <telerik:GridViewDataColumn.Header>
      <TextBlock Text="Name" telerik:LocalizationManager.ResourceKey="title_Name" />
   </telerik:GridViewDataColumn.Header>
</telerik:GridViewDataColumn>


<telerik:RadRibbonButton Text="Duplicate" CollapseToSmall="WhenGroupIsMedium" Size="Medium"
     LargeImage="/Prodigy.UI.RDPInfrastructure.Silverlight;component/Resources/Images/Ribbon/Clone.png"
     Command="{Binding DuplicateCommand}" />

<telerik:RadRibbonButton Text="{Binding LocalizationManager[act_Duplicate]}" Size="Medium"
     LargeImage="/Prodigy.UI.RDPInfrastructure.Silverlight;component/Resources/Images/Ribbon/Clone.png"
     Command="{Binding DuplicateCommand}" CollapseToSmall="WhenGroupIsMedium" />
```

Obrázok č.5: Príklad vloženia lokalizácie.

Následne som pridal preklad vybraných elementov pomocou príkazu telerik:LocalizationManager.ResourceKey=”prefix_vyraz” a použitý výraz do databáze. Aplikáciu som reštartoval a sledoval, prejavenia zmien na UI v časti na ktorej som aktuálne pracoval. Po vizuálnej kontrole prekladu som sa postupne prepracovával k ďalším View

v ďalších častiach aplikácie. Pri práci na lokalizácii RD-Processu som prehliadol viac ako 240 View tried, pričom priemerne každá trieda obsahovala 150 riadkov zdrojového kódu Silverlight.

Pri riešení druhej úlohy som konzultoval vývoj Classification modulu so softvérovým architektom, ako aj mojim vedúcim a inými vývojármi. ProductClassification je entita, ktorá reprezentuje tabuľku uloženú v databáze. Tato tabuľka má atribúty id, name, code, description, idClassificationLevel, insertedBy, insertedOn, updatedBy, updatedOn, idParent. Bola mi načrtnutá základná funkcionalita a nahľaď na dizajn UI. Užívateľské rozhranie obsahuje strom, zobrazujúci hierarchiu vlastných kategórii, detail s obsahom základných informácií a vlastnú listu s príkazmi viz obrázok č.6. Prekonzultovali sme logiku pridávania položiek do stromovej štruktúry a závislosti pri priradzovaní stupňa klasifikácie jednotlivým položkám v rámci jedného stromu. Bol my vysvetlený problém kontroly pred ukladaním a mazaním jednotlivých položiek z pohľadu závislosti rodič – dieťa.

Prvým krokom bolo vytvorenie a začlenenie modulu s názvom Classification v záložke Maintenance. Tento modul som importoval do aplikácie pomocou triedy RDPInfrastuctureModule, v ktorej som vytvoril potrebnu IMaintenanceItem.

Ďalšia potrebná časť bola implementácia triedy ProductClassificationViewModel. V tejto časti nastala zmena v implementácii, ktorá sa používala doteraz. To znamenalo že doteraz sa dáta zobrazované vo View priamo odkazovali na entitu, pričom novy spôsob zamedzil priamemu priestupu View k entite a komunikuje iba s ViewModelom používanej entity. ProductClassificationViewModel implementuje EntityViewModel, takže bolo potrebne dopísať abstraktne metódy Create(), Save(), Delete() a GetDetail() ako aj vlastnosti, ktoré používa entita ProductClassification ako je Title, Name, Code, Description, IdProductClassification a ProductClassificationLevel.

Ako ďalší krok bola implementácia triedy ProductClassificationTreeViewModel a ProductClassificationTreeView. ViewModel obsahuje celu logiku použitú k vytvoreniu stromu a View implementuje okno pre tento strom, na ktorého vytvorenie sa používa komponent od spoločnosti Telerik s názvom RadTreeView. Po vytvorení a spustení UI časti s prázdnym stromom som potreboval implementovať metódy používajúce Enterprise Service Bus. Tieto metódy slúžia k tomu aby som mohol načítať dáta z medzi pamäte Repository pripadne túto Repository doplní o aktuálne dáta z databáze. Následne som v projekte ProdigyEsbModulesRdp implementoval SetProductClassification(ProductClassification), GetProductClassificationByParent(long?) a DeleteProductClassification(ProductClassification).

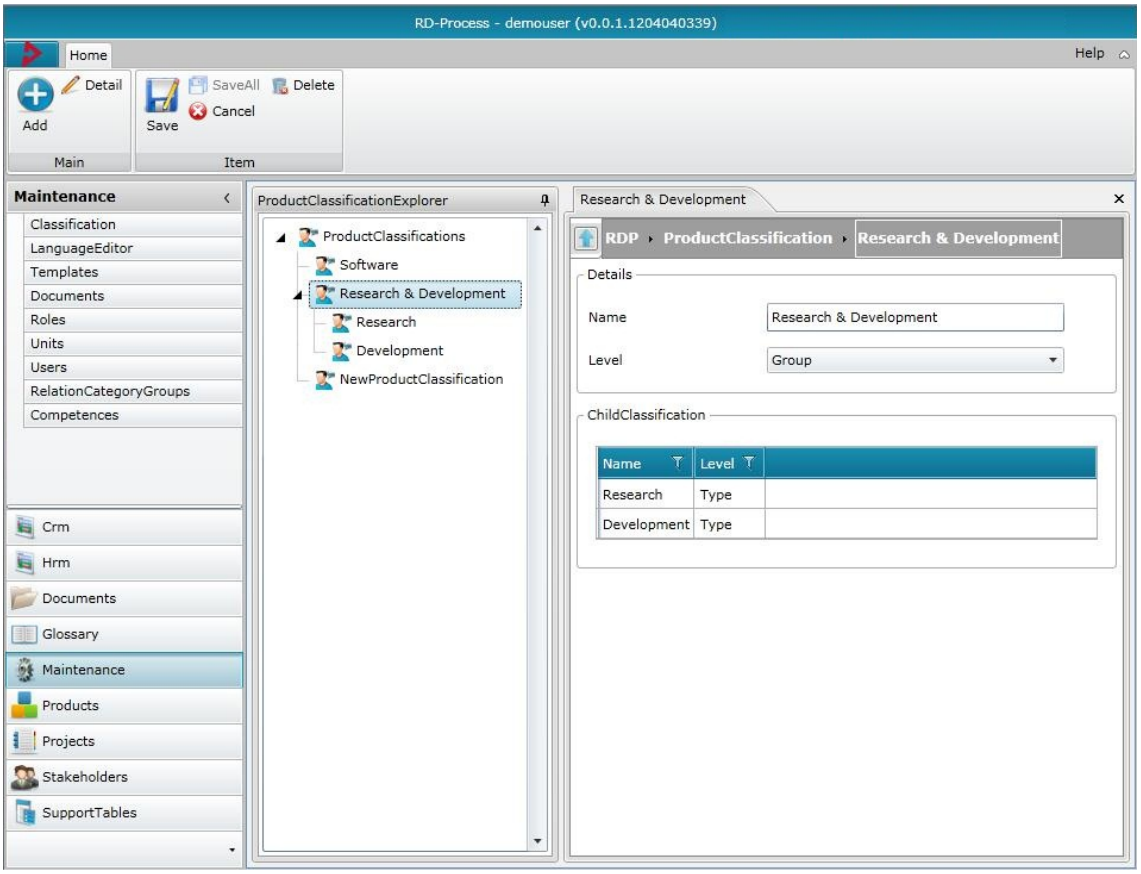
Vytvoril som triedu ProductClassificationTreeItemViewModel, reprezentujúcu jednu položku v strome. Súčasťou triedy je metóda používaná na načítavanie záznamov, ktoré sú v hierarchií pod touto položkou. V tejto metóde používam volanie ESB metódy, na načítanie informácií o deťoch aktuálnej položky v strome.

Následne som si v triede ProductClassificationTreeViewModel vytvoril metódu na načítanie koreňových položiek stromu. Položky stromu sú koreňové pravé vtedy, ak majú nastavený atribút idParent na null. Obdobne ako pri načítavaní informáciách o deťoch som použil ESB metódu, na načítanie koreňových záznamov. Po načítaní potrebných entit je vytvorená štruktúra stromu a tieto entity sú predane ďalšej metóde, ktorá ich inicializuje a pridá do stromu ako RootItem.

Ďalšia potrebnou časťou bolo vytvorenie ProductClassificationOverviewViewModel a View, kde sa zobrazujú koreňové položky v tabuľke spolu zo všetkými dostupnými informáciami. ProductClassificationDetailViewModel a View na zobrazovanie detailných informácií položiek v strome. V tejto časti som sa zaoberal aj navigáciou, aby sa po označení položky v strome zobrazil detail tej istej položky. K riešeniu tohto problému som použil vlastnosť Prismu, ktorý pri načítaní aplikácie exportuje všetky inštancie ViewModelov každej položky v strome. Tieto položky sa dajú v potrebnej chvíli znovu načítať pomocou jednoznačnej identifikácie Guid a modulu Entity Manager, ktorý obsahuje metódu na ziskanie alebo vytvorenie potrebneho ViewModlu pomocou Guid. Takže v ProductClassificationDetail ViewModely sú nastavene vlastnosti ktoré slúžia k zobrazovaniu detailu ako je Name, Level a metoda LoadChildren() a LoadItems(), ktoré načítavajú informácie o deťoch vyuzivane k zobrazeniu v tomto detaily.

Po úspešnom načítaní a zobrazovaní dát uložených v databáze je potrebne implementovať lištu s operáciami Add, Detail, Save, Delete, Cancel. Ako prvé som vytvoril triedu ProductClassificationCommands, ktorá obsahovala všetky potrebne príkazy. Potom nasledovalo vytvorenie ProductClassificationRibbonTabView a ViewModel. Takto vytvorene príkazy som použil vo ViewModeloch na volanie kľúčových metod Create(), Delete(ProductClassification), Save(ProductClassification) a NavigateToDetail(). V týchto metódach je implementovaná cela logika potrebná k bezpečnému vytváraniu, upravovaniu a mazaniu položiek v strome a z toho vyplývajúce zmeny na databáze. Jedna s posledných vecí, ktorú som spolu s kolegom začleňoval do môjho modulu bol Breadcrumb. Je to komponent ktorý slúži k inému spôsobu navigácie medzi položkami v strome. Po vlastnom otestovaní

modulu Classification a odstránení väčšiny chýb som dal check-in a posunul tento modul do testovania. Všetky zdrojové kódy sú uvedené v prílohe č.2. V realizácii úlohy vývoja modulu Classification som napísal 16 tried ktoré obsahujú viac ako *2000 riadkov* zdrojového kódu.



Obrazok 6: Screenshot Product Classification modul

Ďalšie úlohy ktoré mi boli prideľované súviseli buď, s chybami prejavujúcimi sa pri testovaní modulu Classification, alebo to boli rôzne menšie zmeny funkcionality, či odstraňovanie nepoužívaných časti aplikácie. Jedna z chýb, ktorá sa vyskytla pri testovaní modulu Classification bolo neošetrenie výnimky pri vymazávaní položky použitej v niektorom produkte. Takže keď si používateľ v produkte vybral vlastnú klasifikáciu, a vrátil sa do modulu Classification kde tuto klasifikáciu zmazal, tak sa vymazala zo stromu ale nie z databáze. Toto bol problém, lebo klasifikácia ktorá je použitá nesmie byt nikdy zmazaná. Pri riešení tohto problému som odhadol situáciu, v ktorej sa problém vyskytol. Zistil som, že pri vymazávaní sa

ako prvá zavolá metóda Delete() na požadovanú položku v strome, ktorá sa vykonala úspešné a po nej nasledovalo odstránenie s databázy, ktoré však vyvolalo výnimku z dôvodu závislosti s tabuľkou Product a prerušilo proces mazania. Riešenie bolo použiť metódu Subscribe() na zistenie spiatočnej odpovede o stave požadovanej operácie na databáze.

```
/// <summary>Deletes this instance.</summary>
/// <returns>Return collection.</returns>
/// <remarks>Author: PavelR @ 1/25/2012 11:39 AM</remarks>
public override IObservable<Unit> Delete()
{
   string entityName = Model.Name.ToString();
   string message = string.Format(CultureInfo.CurrentCulture, "{0}  has been successfully deleted.", entityName);
   IObservable<Unit> result = null;
   if (Model.Id != 0)
   {
      result = RDPSupportTablesRepository.DeleteProductClassification(Model);
      result.Subscribe(
         next => ToastManager.SendMessage("Delete successfull", message, ToastMessageType.Info),
         ex => { },
         OnDelete);
   }
   else
   {
      ToastManager.SendMessage("Delete successfull", message, ToastMessageType.Info);
      OnDelete();
   }
   return result;
}
```

Obrázok č.7: Použitie metódy Subscribe()

Metoda Subscribe(next,exception,onComplet) zachytáva a spracováva výsledok vrátený službou, ktorá sprostredkovala operáciu delete. Operácia next sa prevedie hneď po úspešnom skončení metódy Subscribe(). Ak pri vykonávaní operácie dôjde k vyvolaniu výnimky prevedie sa akcia exception, ktorá neurobí žiadnu operáciu a ak je operácia potvrdená tak sa prevedie akcia onComplet, ktorá spustí vymazanie položky v strome. Riešenie spočíva aj v kontrole či model.id neje rovne nula. Tato situácia nastane, keď je instancia vytvorená, ale nie je uložená v databáze. Pri tejto situácii stačí vymazať záznam iba zo stromu. Pri použití metódy Subscribe() sa záznam zo stromu výmaze vždy až po úspešnom vymazaní z databázy.

4. Prínosy bakalárskej praxe

Bakalárska prax mi ukázala ako sa pracuje v tíme pri vývoji veľkého projektu. Poskytla mi náhľad na reálne fungovanie v tíme, ktorý používa metodiku Scrum k vývoju svojho projektu. Každodenný míting pôsobí motivačne, a ukazuje celkový náhľad na postupé vyvíjanie projektu, ku ktorému sa môže každý člen tímu vyjadriť a spoločne riešiť problémy vo vývoji. Bakalárska prax mi taktiež ukázala, ako funguje vývoj na zdieľanom kóde pomocou TFS2010 a využívanie funkcií ako GetLatest, CheckOut, Check-in poprípade zjednocovanie dvoch verzií jedného zdrojového kódu. Cely zdrojový kód, ktorý som implementoval bol na vysokej úrovni a musel spĺňať Prodigy Programming Quality Standards, podľa ktorých bola nastavená politika nastrojov CodeAnalysis a StyleCop.

4.1. Teoretické znalosti uplatnené v priebehu praxe

V priebehu svojho štúdia som absolvoval predmety, ktoré sa zaoberali programovaním v jazyku C#, ako aj predmety zamerané na algoritmické myslenie. Pri programovaní v škole sme používali Microsoft Visual Studio 2010, v ktorom som sa naučil pracovať s referenciami a nástrojom na debugovanie aplikácie krok po kroku. Tieto teoretické znalosti slúžili ako dobrý základ pri práci na zadaných úlohách. Taktiež som absolvoval predmety zaoberajúce sa databázovými nástrojmi ako je Microsoft SQL Management Studio. V tomto nástroji som pridával položky do tabuľky prekladov, alebo záznamy do tabuľky ProductClassification aby som videl, či funguje správne načítavanie záznamov.

4.2. Zručnosti získané v priebehu prax

V priebehu praxe som získal väčšie skúsenosti s programovaním v jazyku C# a .Net. Naučil som sa pracovať s rôznymi rozšíreniami ako aj s entitami alebo službami. Pri riešení menších úloh som zdokonalil svoju schopnosť identifikovať situáciu a časť kódu v ktorej aplikácia zlyhávala, čo pomohlo rýchlejšiemu riešeniu problému. Spolupráca na vývoji aplikácie pomocou MVVM poskytuje komplexnejší náhľad na riešenie vývoja dynamických aplikácií. V spoločnosti Stratech som po prvý raz pracoval s programovacím jazykom Silverlight, ako aj jeho nadstavbou poskytovanou firmou Telerik. Skúsenosti s praktickým zapojením sa do vývoja softvéru zo všetkým čo k tomu patri, mi poskytuje veľký náskok pri uplatnení v nasledujúcom profesnom uplatnení.

4.3. Celkové zhodnotenie

Z mojeho pohľadu celkové zhodnotenie bakalárskej praxe v spoločnosti Stratech je kladné. Celý vývojový tím v ostravskej pobočke bol priateľský a pri riešení mojich problémov mi pomáhali svojimi radami a pokročilými znalosťami architektúry Prodigy. Nepracoval som tak efektívne ako ostatný vývojári, čo bolo zapríčinené absenciou praktických skúseností, ako aj nekontinuálnou prácou dva dni v týždni, alebo striedaním sa na jednom počítači s ďalším študentom. V pracovnom počítači sme mali spoločný pracovný priestor (Workspace), v ktorom sú uchovávané všetky zdrojové kódy. Naša individuálna práca na rôznych úlohách ovplyvňovala spúšťanie a prácu na RD-Processe. Tato skutočnosť mi rozvinula schopnosti diagnostiky a odstraňovania problémov. Naučil a zdokonalil som sa vo veľa smeroch programovania, ako aj celkového procesu vývoja softvéru, ktoré sú cennými skúsenosťami hodnotenými pri výberových kritériách v získavaní práce.

5. Literatúra

[1] Stratech Software. [online]. 2012 [cit. 2012-04-13]. Dostupné z: www.stratech.nl

[2] QUINTEN, Vincent. STRATECH SOFTWARE. *PDY-MI-39–Prodigy Architecture Guide*. 2011, 73 s.

[3] MACKEY, Alex. *Introducing .NET 4.0 with Visual studio 2010*. New Edition. Berkeley, CA: Apress, Inc, 2010. ISBN 14-302-2455-X.

[4] IAN GRIFFITHS, Matthew Adams. *Programming C# 4.0*. 6th ed. Beijing: O'Reilly, 2010. ISBN 978-059-6159-832.

[5] BROWN, Pete a Chad A CAMPBELL. *Silverlight 4 in action*: *Silverlight 4, MVVM, and WCF RIA Services*. Greenwich: Manning, 2010, 766 s. ISBN 19-351-8237-4.

[6] LÖWY, Juval. *Programming WCF services*. 3rd ed. Sebastopol, CA: O'Reilly, 2010. ISBN 978-059-6805-487.

[7] BRUMFIELD, Bob. *Developer's guide to Microsoft Prism 4*: *building modular MVVM applications using Windows Presentation Foundation and Microsoft Silverlight*. Redmond, Wash: Microsoft, 2011, 261 s. ISBN 07-356-5610-X.

6. Prílohy

A. Obsah CD

1. Vlastné vypracovanie bakalárskej prace
2. Príloha č.1
3. Príloha č.2